Asymptotic Quantization: A Method for Determining Zador’s Constant

Joyce Shih

Introduction

With the growth of the internet and digital media, data compression is becoming increasingly important due to the overwhelming abundance of information that computer users wish to transmit and store. Quantization refers to the methods by which analog signals are converted into digital representations and compressed, thereby making them suitable for storage. The asymptotic optimal performance of vector quantizers of fixed dimension and large rate was first developed in a rigorous fashion by Paul Zador. This paper describes a Lagrangian formulation of Zador’s quantization results and applies it to estimate Zador’s constant. Knowledge of Zador’s constant may improve current quantizer design techniques by providing theoretical performance bounds.
A vector quantizer system, q, is comprised of two functions, an encoder and a decoder, which are designed using the Lloyd Algorithm. The encoder \( \alpha \) examines the input source X, blocks it into vectors of length \( k \), and maps each vector onto a codeword in the codebook, C. The codeword is then sent through the channel, after which the decoder \( \beta \) maps the codeword onto its corresponding reproduction vector.

The encoder \( \alpha \) can be decomposed into two parts, \( \alpha = \gamma \circ \alpha \). In the first step, \( \alpha \) maps the signal input onto the reproduction vector in the codebook that best matches it. In the second step, \( \gamma \) converts the reproduction vector into a codeword. Similarly, the decoder \( \beta \) can be broken down into \( \beta = \gamma \circ \beta \), where \( \gamma \) associates the codeword with its corresponding vector in the codebook, and \( \beta \) outputs the reproduction vector. The coder \( (\alpha, \beta) \) can thus be rewritten as \( (\alpha, \gamma, \beta) \). This is represented pictorially in Figure 1.

The quality of a k-dimensional vector quantizer q can be measured in terms of its distortion, which quantifies the loss of information resulting from approximating X, the input source, as \( Y \), the code (reproduction) vector. For the sake of simplicity, we take the distortion to be the mean-squared difference:

\[
d(X, Y) = \| X - Y \|_2^2 = \sum_{i=1}^{k} | X_i - Y_{i}\|_2^2
\]

between the input source X and the reproduction vector \( Y \), where the subscript \( i \) refers to the \( i \)th reproduction vector in the codebook, \( I \) is the number of bits used to represent each source vector, and \( k \) is the dimension. Typically we are interested in the performance of an algorithm given a variety of input signals. We thus define an average distortion:

\[
D_j(q) = \sum_i \int_{S_j} f(x) \| X - Y_i \|_2^2 dx
\]

where \( f(x) \) is the probability distribution function that describes the likelihood that the source X will correspond to a given reproduction vector \( x \), and \( S_j \) is the \( j \)th codeword (i.e., the codeword that corresponds to the reproduction vector \( x \)).

While distortion measures the fidelity of a quantizer, rate measures its "cost:" the number of bits required to express the codeword for transmission to the decoder via the channel. The average rate of a quantizer is given by

\[
R_j(q) = \sum_i p_j(S_i) l(i),
\]

where \( p_j(S_i) \) is the probability of using the \( j \)th codeword \( S_i \), and \( l(i) \) is the "cost" (i.e., length in bits) of the \( j \)th codeword.

The optimality of a quantizer is determined by both its distortion and rate (i.e., its quality and cost), which can be expressed as

\[
\rho(f, \lambda, q) = D_j(q) = \lambda R_j(q).
\]

where \( f \) is the probability distribution function (p.d.f) which characterizes the input source and \( \lambda \) is a Lagrange multiplier which quantifies the importance of rate relative to distortion for a given application. If \( \lambda \) is small, it allows a large rate (i.e., high cost or long codeword length), so a larger codebook can be used and the distortion introduced by compression will be minimal. Conversely, if \( \lambda \) is large, the rate must be small, because cost is critical. In this case, a high degree of compression is key, and quality of the reproduction (i.e., information loss or distortion) is less important. For the purposes of this paper, we will limit ourselves to considering systems in the large rate limit (i.e., small values of \( \lambda \) and little distortion).

The classic approach to describing optimal performance is in terms of the distortion-rate function. For rate greater than zero (i.e., \( R > 0 \)), the operational distortion-rate function is defined as \( \delta_{j}(R) = \inf_{q: R_j(q) = R} D_j(q) \). Zador proved that under certain conditions on the p.d.f. \( f \),

\[
\lim_{R \to \infty} 2^{2k} \delta_j(R) = b_{2,k} 2^{h(f)},
\]

where \( b_{2,k} \) is Zador's constant, which depends only on \( k \) and not on \( f \), and \( h(f) = \int f(x) \log f(x)dx \) is the differential entropy of \( f \). While the exact value of \( b_{2,k} \) is known for \( k = 1 \), only upper and lower bounds are known for higher dimensions (although it is known that it converges as \( k \to \infty \)). Therefore, the goal of this project was to compute values for Zador's constant (i.e., \( b_{2,k} \)) for dimensions greater than one by writing a computer program to perform entropy-constrained vector quantization (ECVQ) simulations. A summary of the quantizer terms is presented in Table 1.

The ECVQ Algorithm

The ECVQ algorithm employed in this research uses the Lloyd algorithm mentioned earlier to design vector quantizers with the least possible distortion subject to a constraint on rate or entropy. Unlike other quantization algorithms, ECVQ jointly optimizes both the rate \( R \) and distortion \( D \) rather than optimizing each quantity separately. ECVQ works by minimizing the Lagrangian functional,

\[
\rho_\lambda(\alpha, \beta) = \mathbb{E}\left[ d(X, \beta(\alpha(X))) + \lambda \big| R(\alpha(X)) \big| \right],
\]

where \( \beta(\alpha(X)) \) is the overall
quantization operation on the input source X, to find the optimal coder. At the cost of higher complexity, ECVQ generally outperforms other entropy-coded quantization schemes, including the scalar uniform threshold, lattice, and constrained number-of-indexes vector quantization schemes.

The ECVQ algorithm consists of four main steps. In the first step, the algorithm obtains an initial reproduction codebook or input source. Since Zador's constant is independent of the distribution, simulations were first performed using the simplest possible nontrivial distribution, a uniform distribution on the k-dimensional unit cube, which puts equal weight on integers between 0 and 1. Later simulations were also performed using a Gaussian distribution,

\[ f = e^{-\frac{(x-u)^2}{2\sigma^2}}, \]

where \( u \) is the mean and \( \sigma \) is the standard deviation, in order to verify that the results were in fact independent of the distribution used.

The second step in the ECVQ algorithm involves training the codebook for each value of \( \lambda \). For decreasing values of \( \lambda \), the algorithm was run until it met a stopping criterion,

\[ \left( \frac{\rho_{\text{old}} - \rho}{\rho} > 0.005 \right). \] (7)

In the third step, the encoder maps each input vector onto the nearest codeword in the codebook by finding the codeword that minimizes \( \rho \) (and hence the distortion):

\[ \alpha(X) = \arg\min_{i \in \mathcal{I}} \left| d(X, \beta(i)) + \lambda \right| R(i), \] (8)

Equation (8) is analogous to nearest neighbor encoding in standard vector quantization (VQ). However, in ECVQ the rate for the particular codeword chosen is updated by

\[ \| R(i) \| = \log_2(1/p(i)), \] (9)

where \( p(i) = P(\alpha(X) = i) \). The final step requires the decoder given by

\[ \beta(i) = \arg\min_{Y \in \mathcal{I}} E \left[ d(X, Y) \mid \alpha(X) = i \right], \] (10)

to compute the conditional expectation of the distortion between the output and the input, given that the encoder produced index \( i \). It effectively computes an average or centroid of all the vectors mapped to a particular cell (i.e., codeword) thus far. This centroid will continue to evolve as more data is mapped to each of the cells. The ECVQ algorithm is summarized in Table 2.

**Computing Zador’s Constant**

We say that a probability distribution function \( f \) exhibits the **Lagrange-Zador property** if the following limit exists:

\[ \lim_{\lambda \to \infty} \left( \frac{\rho(f, \lambda)}{\lambda} + \frac{k}{2} \ln \lambda \right) - h(f) = \theta_k, \] (11)

where \( \theta_k \) depends only on the dimension, not on the p.d.f. The Zador constant, \( b_{2,k} \), and \( \theta_k \) are related by

\[ \theta_k = \frac{k}{2} \ln \frac{2e}{k} b_{2,k}. \] (12)

Therefore,

\[ \theta_k = \frac{k}{2} \ln \frac{2e}{k} b_{2,k} = \frac{D_k(q) + R_k(q) - h(f)}{\lambda} + h(f) + \frac{k}{2} \ln \lambda, \] (13)

For the special case of a uniform distribution function, \( h(f) = 0 \). Examining Equation (13), it can be seen that it is possible to calculate \( b_{2,k} \) if the values of \( D_k \) and \( R_k \) are first obtained via computer simulations running the ECVQ algorithm.

**Results**

Estimates of Zador’s constant obtained from computer simulations are reported in Table 3 for \( k = 1 \) through 4. For comparison purposes, the upper and lower bounds for dimensions two through four, along with the exact value of Zador’s constant for a one-dimensional system, are also given. Because Zador’s constant is independent of the probability distribution function, the simulations were run using a uniform distribution for simplicity. In performing the simulations, both the codebook size and the number of training vectors were varied as detailed below.

For dimensions 1, 2, and 3, the codebook size was held constant at 1024. For dimension 1, the test was run five times with 50,000 training vectors, once with 100,000 training vectors, and three times with 250,000 training vectors. The average value obtained for \( b_{2,1} \) from these simulations was 0.08323, which deviates from the actual value by 0.1%. Because algorithm performance improves as codebook size increases, we focused on 250,000 training vectors for the second dimension. The test was run seven times with 250,000 training vectors and once each with 50,000, 100,000, and 500,000 training vectors. The tests gave an average value of 0.07918 for \( b_{2,2} \), which differs from Zador’s constant for fixed rate coding by 1.3%. Although it is not known whether the values of \( b_{2,k} \) for the fixed rate and variable rate cases are identical, it has been conjectured that the two constants are the same. For \( k = 3 \), eight simulations using 250,000 training vectors and two simulations using 500,000 training vectors were run. The results gave an average value of 0.079 for \( b_{2,3} \), which falls within the known upper and lower bounds.\(^5\)
For dimension 4, seven simulations using 250,000 training vectors and another three simulations using 500,000 training vectors were run with a codebook size of 1024. In addition, a single simulation was run utilizing 500,000 training vectors and 2048 codewords. The average value of $b_{2,4}$ obtained from these simulations was 0.07776, which again falls within the known upper and lower bounds.\(^5\)

As mentioned earlier, Zador's constant is independent of the probability distribution function. However, to check that the simulation results were in fact independent of the probability distribution function used, simulations were also performed for the first dimension using a Gaussian distribution function. Ten tests were run using 250,000 training vectors and 1024 codewords. The value obtained for $b_{2,1}$ from these simulations was 0.0835, which agrees well with both the actual value of 0.0833 and the value of 0.08323 obtained from the simulations performed using a uniform distribution function. Thus, it appears that the choice of a uniform distribution did not bias the results.

For the infinite case $b_{2,\infty}$ can be calculated by using Zador's result,\(^4\)

$$\frac{1}{1 + \beta} V_{\beta} \leq kb_{2,1} \leq \Gamma(1 + \beta)V_{\beta}, \quad (14)$$

where $\beta = \frac{r}{k}$ and $r = 2$ for mean squared error; $\Gamma(x)$ is the gamma function, and $V_{\beta}$ is the volume of a unit sphere in $k$ dimensions. The infinite case shows $b_{2,k}$ converges upon the value 0.05854.

Discussion

As stated above, the goal of quantization is to convert continuous signals into bits in a way that optimally trades off distortion or signal to noise ratio (SNR) with bits. Quantization theory provides quantitative relations between distortion and bit rate under certain assumptions. For example, the famous "6-dB-per-bit-rule" describes how the SNR for a uniform scalar quantizer with high rate and low distortion increases 6dB for each one bit increase of rate. Ideally, we would like to minimize both distortion and bit rate, but each can be decreased only at the expense of increasing the other and hence it is the tradeoff that matters. Zador characterized the tradeoff under quite general conditions for the "high rate" case where the bit rate is high and the distortion small, the situation arising in most modern analog-to-digital converters.

Zador's constant can be viewed as a fundamental constant of nature. It describes the relationship between distortion and rate in a manner similar to the way $\pi$ describes the relationship between the circumference and radius of a circle. However, unlike $\pi$, the exact value of Zador's constant is only known in the first dimension. Estimating the constant in higher dimensions is of interest because it allows the application of theoretical results to predict the performance of vector quantizers on real world signals.

Conclusion

Quantization is in essence analog to digital conversion for the purpose of storage in a digital channel. Quantization is becoming increasingly essential as digitization and the internet require improved methods of conserving memory and storage space. Starting with the Lagrangian formulation of Zador's results, the generalized Lloyd ECVQ algorithm has been employed to estimate Zador's constant for $k = 1$ through 4 in hopes that knowledge of Zador's constant may lead to improved quantizer design techniques.

Works Cited


Figure 1: A Communication System

Table 1. Summary of Quantizer Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distortion</td>
<td>Quantifies the difference between the input source and the reproduction vector. Distortion measures the fidelity of the quantizer.</td>
</tr>
<tr>
<td>Rate</td>
<td>Rate measures the cost or number of bits required to express a codeword.</td>
</tr>
<tr>
<td>Optimality</td>
<td>The optimality of a quantizer depends on its distortion and rate. This value can be expressed as $\rho(f, \lambda, q) = D_f(q) + \lambda R_f(q)$, where $\lambda$ quantifies the importance of rate relative to distortion.</td>
</tr>
</tbody>
</table>
Table 2. ECVQ Algorithm

<table>
<thead>
<tr>
<th>Steps</th>
<th>Description</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Obtain an initial reproduction codebook.</td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>Train the codebook for each $\lambda$.</td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td>The encoder maps each input vector onto the nearest codeword in the codebook that minimizes the cost.</td>
<td>$\alpha(X) = \arg\min_{i\in I} [d(X,\beta(i)) + \lambda</td>
</tr>
<tr>
<td>(4)</td>
<td>The decoder computes the centroid for each codeword or cell based upon all the input vectors mapped onto that codeword or cell.</td>
<td>$\beta(i) = \arg\min_{Y\in Y} E[d(X,Y)</td>
</tr>
<tr>
<td>(5)</td>
<td>Repeat steps 1-4 until stopping criterion has been reached.</td>
<td>$(\frac{\rho_{old} - \rho}{\rho} &gt; 0.005)$</td>
</tr>
</tbody>
</table>

Table 3. Values and Bounds for Quantization Coefficients $b_{2k}$

<table>
<thead>
<tr>
<th>$K$</th>
<th>Sphere Lower Bound</th>
<th>Actual Value</th>
<th>Simulation Value</th>
<th>Cube Upper Bound</th>
<th>Zador Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.08333</td>
<td>0.08333</td>
<td>0.08323</td>
<td>0.08333</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>0.07958</td>
<td>0.07918</td>
<td>0.08333</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.07697</td>
<td>0.07900</td>
<td>0.08333</td>
<td>0.1157</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.07503</td>
<td>0.07776</td>
<td>0.08333</td>
<td>0.09974</td>
<td></td>
</tr>
<tr>
<td>$\infty$</td>
<td>0.05854</td>
<td>0.05854</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>